EUV IMAGING SPECTROMETER

Hinode

EIS SOFTWARE NOTE No. 16

Version 2.5 6 January 2015

EIS_AUTO_FIT and SPEC_GAUSS_EIS: Gaussian fitting routines for the Hinode/EIS mission

Peter Young
George Mason University
4400 University Drive
Fairfax, VA 22030
U.S.A.

pyoung9@gmu.edu

A basic requirement for any emission line spectroscopy mission is a set of software suitable for fitting Gaussians to lines in order to derive intensity, velocity and line widths. This document describes a set of IDL routines available for the Hinode/EIS mission that are intended to cover most situations encountered by EIS observers.

Since EIS yields spatially-resolved data that often have sufficiently high signal-to-noise to yield good line profiles in individual spatial pixels, then a common requirement is for a fitting routine that automatically goes through a raster and fits Gaussians at each pixel. We refer to this as Case 1.

The second scenario is when a user wants to obtain a spectrum for a particular spatial feature that spans more than one pixel. Examples include bright points, active regions, flares, quiet Sun and coronal hole. In this case the user wants a single spectrum that has been averaged over many pixels. The situation is complicated with EIS as spatial offsets and mis-alignments within the instrument lead to the spatial feature occurring at different locations in Y on the detector depending on the wavelength. The Gaussian fitting routine in this case will be a manual one (since there is only one spectrum) and should allow the user to sequentially go through the spectrum fitting lines individually or through multi-Gaussian fits. This scenario is referred to as Case 2.

The routines described in the following sections deal with the following situations:

Case 1

- both single and multi-Gaussian fitting (automatic)
- correcting for the slit tilt and orbit variation
- selecting sub-regions within a wavelength window for fitting
- specifying initial parameters and parameter value limits

Case 2

- correcting for spatial offsets as a function of wavelength
- correcting for slit tilt and orbit variation
- choose a pixel mask to identify feature within a raster
- a manual single and multi-Gaussian fitting routine

A key starting point for the Gaussian fitting routines is the extraction of an EIS data window into an IDL structure using the routine *eis_getwindata.pro*. We often refer to the *windata* structure in the sections below, which is the output from this routine. There are a number of routines for manipulating the *windata* structures and these are described in the Appendix.

A separate document (EIS Software Note #17) is available with worked examples for the Gaussian fitting routines.

Case 1: automatic Gaussian fitting

The basic fitting routine here is *eis_auto_fit.pro* which deals with both single and multiple Gaussian fitting, however the latter will be dealt with separately in Section 2. There is a varying level of complexity in how the fits are performed, with the velocity requiring the most consideration due to the lack of an absolute calibration for EIS as well as instrumental effects that change the position of the line centroid on the detector. Since an EIS raster may contain many thousands of spatial pixels it is important for the user to be able to assess the quality of the line fits. This is done with the routine *eis_fit_viewer.pro*.

The first step in using the *eis_auto_fit* suite of routines is to extract a single EIS data window into an IDL structure using the routine *eis_get_windata*, e.g.,

```
IDL> windata=eis_getwindata(l1name,195.12,/refill)
```

Generally an EIS data window contains a single emission line, but there are many cases where lines are blended or there are multiple emission lines in a window. The most extreme case is for full CCD spectra where an EIS data window will contain one quarter of the complete EIS spectrum, containing many emission lines.

It is generally recommended to use the /refill option in the call to eis_getwindata. This option is described in EIS Software Note #13.

1 Single Gaussian fitting

At the most basic level, an automatic single Gaussian fit can be performed with, e.g.,

```
IDL> windata=eis_getwindata(l1name,195.12,/refill)
IDL> eis_auto_fit, windata, fitdata
IDL> eis_fit_viewer, windata, fitdata
```

The eis_auto_fit routine is described in Section 1.1, and eis_fit_viewer is described in Section 1.2. In cases where there are multiple lines in the EIS data window, it will be necessary to choose a restricted region of the spectrum before performing the fit. This selection is performed with the routine *eis_wvl_select* and is described in Section 1.3.

1.1 eis_auto_fit

The simplest calling sequence for eis_auto_fit is:

```
IDL> eis_auto_fit, windata, fitdata
```

where the output*, fitdata,* is an IDL structure. For information about how to extract intensity, velocity and line width arrays from this structure, see the routine *eis_get_fitdata* that is discussed in Section 3. A full description of the contents of *fitdata* is given in the header of the *eis_auto_fit* routine.

In order to speed up the fitting process and ensure reasonable fit parameters, eis_auto_fit places limits on the allowed range of the peak, centroid and width of the Gaussian. For each spatial pixel, the routine finds the wavelength pixel in the spectrum with the largest intensity value. The wavelength of this pixel serves as the initial guess of the line centroid, and a range of ± 0.2 Å

around this value is allowed for the fitted Gaussian. The full-width at half maximum (FWHM) is forced to lie between 47 and 235 mÅ, and the line peak is forced to be ≥ 0. The routine keeps track of whether any of these parameter limits are reached, and a summary is printed after eis_auto_fit has completed. The parameter limits are fixed in this basic call to eis_auto_fit, but can be modified by the user by specifying a template for the fit. This is discussed in Section 2.

There are a number of factors that will prevent *eis_auto_fit* from attempting a fit to a spectrum and, if no fit was performed to any spatial pixel, a summary of why the fit was not performed will be printed to the screen. In addition the tag *bad_pix* within *fitdata* will contain one of the following numbers to indicate what happened:

0. Fit was performed.
1. Fit not performed. Spectrum too noisy.
2. Fit not performed. Too many missing pixels.
3. Fit not performed. Spectrum was completely missing.
4. Fit not performed. The wavelength offset value (see Sect. 1.6) was missing.

'Missing' in this case means that a value was set to the official EIS missing value, which is stored in *windata.missing*. A spectrum with intensity values Y_i and error values E_i is deemed to be noisy if max(Y)-median(Y) ≤ 2 x max(E). A spectrum is deemed to have too many missing pixels if the number of fit parameters is ≥ the number of data points being fit.

1.2 Viewing the fits, and obtaining fit parameters

The widget-based routine *eis_fit_viewer* allows the user to study the intensity, velocity and line width maps, and to assess the quality of the fits at individual pixels. It also shows histograms of the fit quantities across the raster. The IDL call is:

```
IDL> eis_fit_viewer, windata, fitdata
```

The top three windows in the GUI contain the intensity map, velocity map and line width map. Text boxes allow the user to specify upper and lower limits for scaling the images.

With the 'Zoom' option selected, you can zoom into any of the three images by clicking-and-dragging the mouse over the windows to create a rubber-band box that selects a region.

With the 'Pixel' option selected, you can click on any pixel in any of the images, and the line profile for that pixel will be shown in the bottom-left graphics window. In the bottom-right window a histogram plot of the selected line parameter (intensity, velocity, width) is shown for the spatial region that is being displayed in the upper windows.

If users need to access fit parameter maps from the IDL command line, then the *eis_get_fitdata* routine is available (see Sect. 3).

1.3 Restricting the wavelength range

EIS studies are usually designed so that a wavelength window contains a single emission line and so the user will simply use the entire wavelength range of the window for fitting. Sometimes, however, a study designer may deliberately have chosen a wide wavelength range to encompass two or more lines, or a line may be very close to another line in the spectrum. In these cases a user can choose either to do a multi-Gaussian fit (see later), or he/she can choose to only fit a subset of the spectral range containing the emission line and the nearby continuum. (Some EIS datasets contain complete EIS spectra and so a single window contains a full spectrum of 1024 pixels. For these the user is recommended to first use the routine *eis_trim_windata* – see

Appendix – to reduce the wavelength coverage of the *windata* structure directly after running *eis_getwindata*.)

To manually select which regions of the spectrum will be included in the fit, the widget-based routine *eis_wvl_select* is used:

IDL> eis_wvl_select, windata, wvl_select

An IDL widget will appear with two graphics windows (see Figure 1). The left panel shows an image derived from the data window (summed over wavelength). Clicking on any position in the image produces a spectrum from that spatial pixel in the right-hand panel. Initially each wavelength pixel in the spectrum has a *. This indicates that the pixel will be included in the fit performed by *eis_auto_fit*. By holding-and-dragging the left mouse button the user can draw a rubber-band box on the spectrum. All wavelength pixels within the X-range of this box (Y-range is not important) will be de-selected, and the * symbols will disappear. This means that those pixels will not be included in the fit. Figure 1 shows the case of a large wavelength window where the pixels from around 263.5 Å and higher have been de-selected. Therefore only the line around 263.0 Å will actually be fit by *eis_auto_fit*.

After selecting a wavelength range, the user can try clicking on different spatial pixels to check if any lines pop up at different locations that could interfere with the fit. When the user is happy with the fit region, clicking 'EXIT' will return the session to the IDL prompt and the selected wavelength region(s) will be stored in the structure *wvl_select*. This structure is then passed to *eis_auto_fit* as follows:

IDL> eis_auto_fit, windata, fitdata, wvl_select=wvl_select

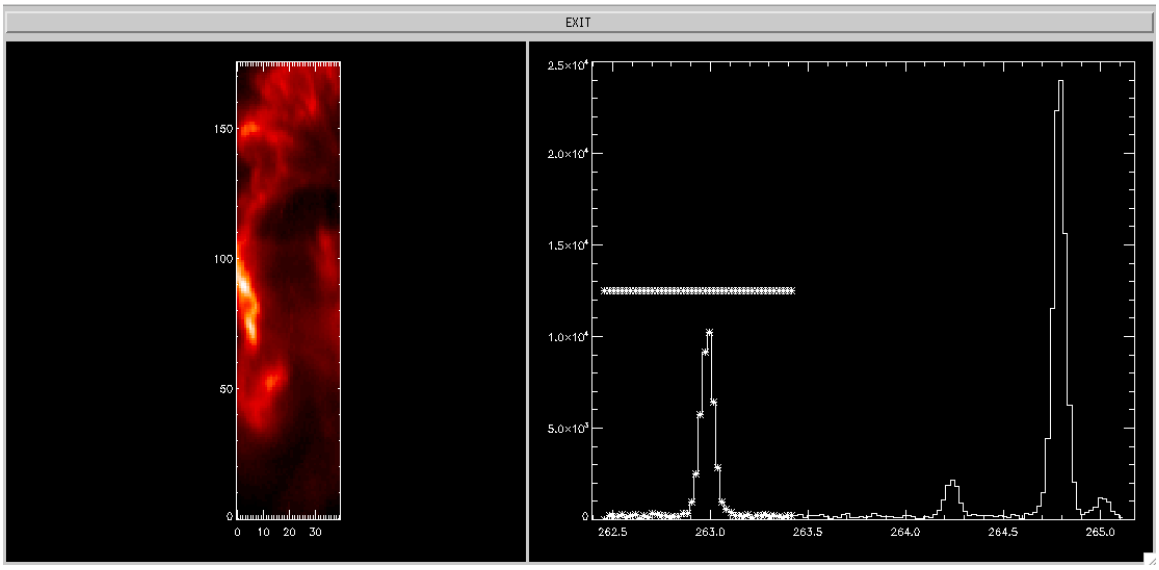

Figure 1 A screen grab of the eis_wvl_select widget. The left panel shows the image derived from the raster, and right panel shows the spectrum from the user-selected spatial pixel. Pixels to be included in the fit are denoted by * symbols on the spectrum.

1.4 Deriving a velocity map from fitdata (the refwvl tag)

A crucial aspect of deriving a velocity map is that a rest wavelength needs to be specified: $v=(\lambda-\lambda_{rest})/c$, where λ is the line centroid and c the speed of light. The velocity map is obtained from *fitdata* through

IDL> vel=eis_get_fitdata(fitdata,/vel)

and the rest wavelength is taken from the *refwvl* tag of *fitdata*. Where does *refwvl* come from?

eis_auto_fit will *always* set *refwvl* to be the average centroid from the data array. For example, if the raster has a size of 100x200 then the line centroids from the fits at each pixel are averaged (note: any 'missing' fits are not included in the average). This definition of *refwvl* means that the average velocity in the raster is set to zero.

The wavelength scale used by *eis_auto_fit* is considered to be an absolute wavelength scale (although see discussion in the following sections) so therefore, if you are confident that you know what the rest wavelength of a line is, *fitdata.refwvl* can be overwritten manually with this rest wavelength. This is especially important for Fe XII λ195.12, which is the reference line by which the wavelength scale is determined (see Section 1.6). For this line the user *must* do:

IDL> fitdata.refwvl=195.119

Some information on how to specify *refwvl* for other emission lines is given in Section 4.

1.5 Common options for using eis_auto_fit

This section summarizes some common options for using the *eis_auto_fit* suite of routines.

Fitting a sub-region of the raster. This can be useful if you are only interested in a sub-region of the raster, and can be done through the xrange= and yrange= optional inputs to *eis_auto_fit*. For example, if you only want to fit the region corresponding to X-pixels 10 to 30 and Y-pixels 100 to 150, then the call is:

IDL> eis_auto_fit, windata, fitdata, xrange=[10,30], yrange=[100,150]

Perform spatial binning of the data before fitting. This is necessary sometimes for weak emission lines in order to improve signal-to-noise. After creating the *windata* structure, the user should call the routine *eis_bin_windata* to perform the spatial binning – see Appendix **Error! Reference source not found.**. The new, spatially-binned *windata* structure can then be used in the same way as the un-binned *windata* structure.

Force the background to be uniform for the fit. When fitting an emission line, *eis_auto_fit* assumes that the background is described by a 1^{st} order polynomial. The user can force the routine to use a 0^{th} order polynomial (i.e., a flat background) by giving the keyword /uniform_backg:

IDL> eis_auto_fit, windata, fitdata, /uniform_backg

Fitting full CCD spectra. When fitting a single line from full CCD spectra, it is recommended that the *windata* structure is first "trimmed" to include only the wavelength region of interest. This is done with the routine *eis_trim_windata* – see Appendix **Error! Reference source not found.**.

1.6 ADVANCED TOPIC: Dealing with the spectrum drift and slit tilt

The centroids of emission lines on the EIS detectors move during the satellite orbit of 98.5 mins by about 2 pixels – a feature known as 'spectrum drift'. In addition, the tilts of the EIS slits mean

that the centroid of a given line can vary by up to 2.5 pixels along the slit. (More information about these instrumental effects can be found on the EIS wiki.)

The most obvious impact of these effects is on velocity maps derived from Gaussian fits, as they will lead to false velocity shifts of 10's of km/s. *The eis_auto_fit suite of routines automatically take care of both the spectrum drift and slit tilt so that velocity maps derived from the line fits will not show their effects*. However, if the user intends to perform scientific analysis of velocity maps then he/she should give some thought to how the spectrum drift, in particular, is corrected.

We first consider how *eis_auto_fit* computes the wavelength array that is used for the line fit. The *windata* structure contains a tag called *wvl* that contains the wavelength vector for the data window. This is a 1D vector that is obtained from the EIS wavelength dispersion relation between wavelengths and pixels, and it does not account for spectrum drift or the slit tilt.

The tag *wave_corr* within *windata* is a 2D array that contains the spectrum and slit tilt corrections for each spatial pixel within the data window. The spectrum drift is computed using the method described by Kamio et al. (2010, Sol. Phys., 266, 209) and the slit tilt is computed from parameters that were also presented in this paper. The Kamio et al. method *yields an absolute wavelength calibration*, and so the quantity:

wvl[x,y] = windata.wvl – windata.wave_corr[x,y]

(where [x,y] is a spatial pixel) gives the absolute wavelength array for this pixel. In this document we generally refer to the 2D array *windata.wave_corr* as the *offset* array.

The routine eis_auto_fit therefore uses the wavelength array wvl[x,y] (as defined above) when performing the Gaussian fits.

The Kamio et al. method derives the spectrum drift for a data-set through a model that makes use of EIS temperature sensors. It does not use the raster's science data in any way. The absolute wavelength calibration is set to force the average centroid of the Fe XII λ195.12 line to be195.119 Å when the entire set of EIS λ195.12 line profiles are averaged over the mission. It is estimated that this yields an accuracy of ± 4.4 km/s (Kamio et al. 2010, Sol. Phys., 266, 209). Alternative methods of deriving an absolute wavelength scale are possible, and an example of how one such method can be implemented through the *eis_auto_fit* suite of routines is described in the next section.

1.7 ADVANCED TOPIC: deriving an alternative absolute wavelength scale

Until the Kamio et al. method was made available in 2010, scientists usually made use of the raster data themselves to derive an absolute wavelength scale for their data. A typical example for active regions is to identify a region of quiet Sun within the raster, and then assume the average velocity shift is zero in this region. This section describes how this method is implemented in the *eis_auto_fit* suite of routines.

The emission line is fit with *eis_auto_fit* in exactly the same way as described in the previous sections:

```
IDL> windata=eis_getwindata(l1name,195.12,/refill)
IDL> eis_auto_fit, windata, fitdata
```

Now consider the case whereby the user determines that the bottom 50 pixels of the raster (pixels 0 to 49) represent quiet Sun. The following command:

IDL> newfitdata=eis_update_fitdata(fitdata,yrange=[0,49])

creates a new *fitdata* structure that has been modified to force the velocity derived from the bottom 50 pixels to average to zero. The *refwvl* tag of *newfitdata* has been modified from that of *fitdata* to be the average centroid over the specified Y-range.

This 'quiet Sun method' is essentially an alternative way of deriving the spectrum drift, and the result can be compared to the Kamio et al. method by doing:

IDL> plot,/ynozero,newfitdata.refwvl[0] – newfitdata.offset[*,0]
IDL> oplot,line=2, fitdata.refwvl[0] – fitdata.offset[*,0]

(The *offset* tag of *fitdata* is identical to the *wave_corr* tag of *windata*.)

If the scientist decides that the quiet Sun method is preferable to the Kamio et al. method, then he/she may want to use the quiet Sun spectrum drift and apply it to other emission lines. This is the subject of the next section.

1.8 ADVANCED TOPIC: running eis_auto_fit with an alternative wavelength offset array

The previous section described how the 'quiet Sun method' could be used to derive an alternative spectrum drift. The routine *eis_update_fitdata* has an optional output called *offset*:

IDL> newfitdata=eis_update_fitdata(fitdata,yrange=[0,49],offset=offset)

offset is an array of same size as *windata.wave_corr* and it serves as an alternative to the latter when running *eis_auto_fit* and associated routines. For example:

IDL> eis_wvl_select, windata, wvl_select, offset=offset
IDL> eis_auto_fit, windata, fitdata, wvl_select=wvl_select, offset=offset

will result in the fit being performed by correcting the wavelength scale with *offset* rather than *windata.wave_corr* (see discussion in Section 1.6).

This is useful when applying the *offset* determined from one line to a different line. Consider the case whereby the quiet Sun method has been applied to Fe XII λ195.12, and the array *offset* has been created (with *eis_update_fitdata*). The next line to be fit is Fe XIII λ202.04, and the new *offset* array can be used:

IDL> eis_auto_fit, windata202, fitdata202, offset=offset

The *refwvl* value for λ202.04 will be set to be the average centroid of the λ202.04 line over the entire raster. Please read Section 4 for more information about how to modify *refwvl* in this case by using the rest wavelength offsets of λ195.12 and λ202.04.

2 Multi-Gaussian fitting

The main difference over single Gaussian fitting is that initial fit parameters for each Gaussian need to be manually specified by the user with a routine called *eis_fit_template*. (For the single Gaussian case, *eis_auto_fit* automatically estimates the initial fit parameters – see Section 1.1.) With this extra step, the sequence of commands to perform multi-Gaussian fitting is:

```
IDL> wd=eis_getwindata(l1name,195.12,/refill)
IDL> eis_fit_template, wd, template
IDL> eis_wvl_select, wd, wvl_select
IDL> eis_auto_fit, wd, fitdata, template=template, wvl_select=wvl_select
IDL> eis_fit_viewer, wd, fitdata
```

Section 2.1 describes how to use *eis_fit_template*

2.1 Creating a fit template

'eis_fit_template' is a widget-based routine and the sequence of operations for using it is are as follows:

1. The left-hand graphic window shows an image derived from 'windata'. Draw a rubber-band box (hold-and-drag the left mouse button) to select a sub-region within the image.
2. A spectrum will appear in the right-hand graphics window. It has been averaged over the spatial region selected in Step 1.
3. Click on the 'Choose lines' button underneath the right-hand graphics window.
4. With single clicks of the left mouse button, click at the approximate locations of the peak of each line you want to include in the fit. A thick vertical line will appear with each click.
5. When you're finished selecting lines, click on the 'End selection' button underneath the right-hand graphics window.
6. You can now go back to the image in the left-hand window and select different spatial regions to see if your original region gives a typical spectrum, or if there are additional lines not apparent in the original region.
7. When you're happy with your line selection, click on the 'Exit' button at the top of the widget.

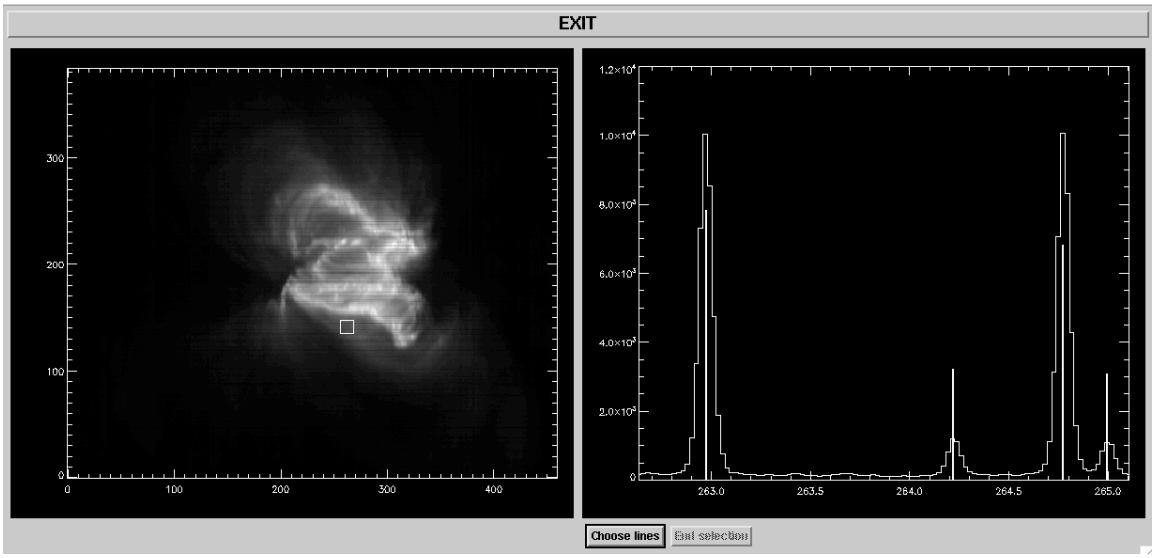

Figure 2 Example of the 'eis_fit_template' widget, showing the EIS image on the left and the spectrum from the selected region (denoted by white square on image) on the right. On the spectrum four vertical lines denote the initial parameters selected by the user.

Some recommendations for using 'eis_fit_template':

1. Over-estimating the peaks of weak lines and under-estimating the peaks of strong lines helps yield better fits.
2. When choosing spatial regions, don't make them too large. Around 10x10 pixels is good. Also making the box bigger in Y is better than making it bigger in X.
3. Try to avoid choosing the brightest region for selecting the template – try to use a weak-to-average brightness region as this will be a better representation of the raster as a whole.

Examples for using eis_fit_template are presented in the accompanying Examples document.

The user-created template is returned to IDL as an IDL structure that can then be input to eis_auto_fit. The template can be saved in the form of a text file by doing:

IDL> eis_write_template, 'template.txt', template

This file can be edited manually be the user, and then read back into an IDL structure by doing:

IDL> template=eis_read_template('template.txt')

This may be useful if the user plans to use a template for many data-sets.

2.2 Viewing the multi-Gaussian fits (eis_fit_viewer)

To view the multi-Gaussian fits produced by 'eis_auto_fit', simply call 'eis_fit_viewer' in the same manner as before:

IDL> eis_fit_viewer, wd, fitdata

The widget shows one significant difference over the single Gaussian version: below the 'Unzoom' button a set of widget buttons is shown giving the 'refwvl' values for each line of the multi-Gaussian fit. By clicking on different lines you will see the images change, giving the intensity, velocity and width for the selected line.

For the spectrum plot (bottom left graphic window), there are also some plot options to make viewing the fits easier. In particular:

- For 'X-range options', clicking on 'Selected line' shows the spectrum for +/- 0.20 Å either side of the selected emission line.
- For 'Y-range options', clicking on 'Selected line' changes the Y-range to better display the selected line.

2.3 Velocities, refwvl and wavelength offsets

For multi-Gaussian fits, the tag *refwvl* – which is used to define the rest wavelengths for velocity work – will be an array with the same number of elements as emission lines in the fit. As described in Section 1.4, the *refwvl* value for each line is calculated by *eis_auto_fit* to be the average of the line centroid over the raster.

Section 1.7 described how the user may choose to use a quiet part of the raster to derive a new absolute wavelength scale. For multi-Gaussian fits, the user must choose which line is used for the quiet region through the line= keyword to *eis_update_fitdata*, e.g.,

IDL> newfitdata=eis_update_fitdata(fitdata, yrange=[0,49], line=0, offset=offset)

Here line 0 is used, and the *offset* output will contain the spectrum drift obtained from line 0 in the quiet region (see Section 1.8). The value of *refwvl*[0] will be updated to be the average centroid of line 0 over the quiet region. The *refwvl* values for the other lines will also be updated by the difference for line 0. For example if the difference between the old *refwvl*[0] value and new one is +0.05 Å, then the *refwvl* values for the other lines will be increased by 0.05 Å as well.

2.4 ADVANCED TOPIC: setting parameter limits

Parameter limits can be useful for preventing bad fits. For example EIS emission lines typically have widths of around 60-80 mÅ. A line cannot have a width of < 50 mÅ as it would be narrower than the instrumental width, while widths of > 200 mÅ are highly unusual. One can thus specify that line widths should lie within the range 50-200 mÅ. Bounds on each of the three emission line parameters (peak, centroid and width) can be specified through the 'template' structure.

'template' has the tag 'lines' which contains a number of tags that apply to each of the Gaussians. The parameter limits are stored in the following tags (for the example of Gaussian 0):

```
IDL> print,template.lines[0].peak_lim
0.00000     -100.000
IDL> print,template.lines[0].cen_lim
256.534      256.834
IDL> print,template.lines[0].wid_lim
0.0200000    0.0900000
```

These are the default parameter limits set by 'eis_fit_template'. For each parameter there are two numbers: the lower and upper limits, respectively. If a limit is set to -100 then it means the parameter is not limited. Therefore the line peak must be ≥ 0, but there is no upper limit. The centroid must lie between 256.534 and 256.834 Å, i.e., ± 0.15 Å of the wavelength selected by the user when using 'eis_fit_template'. The Gaussian width must lie between 0.020 and 0.090 Å. Note that the Gaussian width is related to the FWHM by the factor 2.35 so this range corresponds to a FWHM range of 47-212 mÅ.

By simply editing the numbers in 'template' one can adjust the parameter limits. E.g., suppose you want to remove the upper limit on line widths for Gaussian 4, then you do:

```
IDl> template.lines[4].wid_lim[1]=-100.0
```

If the template structure has been written to a text file using 'eis_write_template' (see Section 2.1) then the limits can be adjusted by opening the template file in a text editor and modifying the lines:

```
Allowed range for wavelength:     256.534     256.834
Allowed range for peak:             0.000    -100.000
Allowed range for width:            0.020       0.090
```

Note that a fixed format of '(f12.3)' is used for these numbers. The new template file can be read into IDL using:

```
IDL> template=eis_read_template('template.txt')
```

2.5 ADVANCED TOPIC: What happens if a parameter limit is reached by the fitting routine?

If a parameter limit is reached at one spatial location, then the 1σ error on that parameter is set to zero and the parameter value is set to the limit. All other parameters will be optimized as normal. *eis_auto_fit* prints a summary of all parameter limits that were reached during the fitting. If there are a lot of such cases, then the user has the following options:

1. Modify the parameter limits in the *template* structure.
2. Bin the data using *eis_bin_windata* and re-do the fit (to increase signal-to-noise).
3. Omit weak lines from the fit using *eis_wvl_select*.

Generally noisy data are the cause of the parameter limits being reached and so options 2 or 3 should be used, however there may be cases where the parameter limits can be tweaked, e.g., if two lines are close together in the spectrum.

If you are fitting flares lines such as Fe XXIV 192.03 or Fe XXIII 263.77, then the default width limits will generally be too small to capture the large broadening often seen for these two lines during flares, and so adjusting the width limits is strongly recommended for these lines.

2.6 ADVANCED TOPIC: tying parameters

In addition to prescribing parameter limits, the user can also tie the parameters of one emission line to another. Three cases are considered:

1. A line has a fixed intensity ratio relative to another line (for example a branching ratio) so the user forces the two lines to have the same width, and their peaks to have a fixed ratio.
2. Two lines are emitted from the same ion so the widths can be set to have the same value.
3. Two lines have a fixed wavelength separation from each other.

The user should generally not use this feature unless the use of fully independent line parameters fails to yield satisfactory results. The accompanying Examples document shows how parameter tying is used for the Fe XII λ195.12+λ195.18, and Fe XII λ203.72+Fe XIII λ203.82 blends.

Parameter tying is implemented through the template structure (Section 2.1). The tags of the 'lines' structure of 'template' include the following:

```
IDL> help, template.lines[0]

PEAK_TIE        INT              -1
PEAK_TIE_VAL    FLOAT          -100.000
CEN_TIE         INT              -1
CEN_TIE_VAL     FLOAT          -100.000
WID_TIE         INT              -1
```

These are the parameters that control the parameter tying. The values of -1 for 'peak_tie', 'cen_tie' and 'wid_tie' indicate that parameter-tying is switched off. Suppose Gaussian 1 is emitted from the same ion as Gaussian 0, and that atomic theory predicts that Gaussian 1 is 0.3 times the strength of Gaussian 0, and occurs at a wavelength 0.32 Å longer than Gaussian 0. These facts can be implemented in the template structure by doing

```
IDL> template.lines[1].peak_tie=0
IDL> template.lines[1].cen_tie_val=0.30
IDL> template.lines[1].cen_tie=0
IDL> template.lines[1].peak_tie_val=0.32
```

IDL> template.lines[1].wid_tie=0

The '_tie' values are set to 0 indicating that the three parameters of Gaussian 1 are tied to the parameters of Gaussian 0. The value 'peak_tie_val' is set to the ratio of Gaussian 0 to Gaussian 1, and 'cen_tie_value' is set to the wavelength offset of the two lines. Note that there is no 'wid_tie_val' as the only option available is to set two lines to have the same width.

The above commands set the parameter tie values for the current session. If you want to set them for multiple data-sets, then you can manually edit the template text file. First write out the template structure to a text file:

IDL> eis_write_template,'template.txt',template

and then open this file in a text editor. For each Gaussian in the file there will be the following text:

```
Parameter tying:
  Amplitude:  -1    -100.000
  Centroid:   -1    -100.000
  Width:      -1
```

where the default values can be manually replaced with the new values. Note that the format is fixed to '(i3,f12.3)'.

To see how the parameter tie values are used by 'eis_auto_fit', try doing:

IDL> expr=eis_fit_function(template)

the resulting string gives the complete fit function that will be used by 'eis_auto_fit' in the call to the minimization routine 'mpfitexpr'. The p[0], p[1], etc., are the free parameters for the fit. Each Gaussian has a corresponding 'gauss_sg' function, and the background is given by either a 'line_sg' function, or a single parameter, depending on if a linear or flat background is specified.

3 Extracting parameters from the fitdata structure

The structure *fitdata* stores the fit parameters in an array of size (*nparam, nx, ny*) under the tag *aa*. It is possible to pick out the peak, centroid and width arrays from this array, however it is easier to use the routine *eis_get_fitdata*. E.g.,

IDL> cen=eis_get_fitdata(fitdata, /cen, line=line)

which extracts the centroids into a 2D array (the optional input line= is used to specify which emission line, in the case of multi-Gaussian fits, is to be extracted). The associated error array can be extracted by using the optional input 'error='. E.g.,

IDL> cen=eis_get_fitdata(fitdata,/cen,line=line,error=cenerr)

Velocities are not stored explicitly in *fitdata*, and so *eis_get_fitdata* is recommended for extracting these:

```
IDL> vel=eis_get_fitdata(fitdata,/vel,line=line)
```

Note that the reference wavelength stored in *fitdata.refwvl* is used to compute the velocities.

By default *eis_get_fitdata* returns the parameter array as a simple 2D image, but by specifying the /map keyword, it will be returned as an IDL map, allowing the wide range of IDL map software to be applied. For example:

```
IDL> intmap=eis_get_fitdata(fitdata,/int)
IDL> plot_map, intmap
```

4 Absolute velocities with EIS

Deriving absolute velocities from EIS data is a complicated issue, and users should take great care in interpreting velocity maps, particularly for features with small velocities of 10 km/s or less. As a general statement, the accuracy of a single EIS velocity measurement will be no better than about 5 km/s. The recommended analysis option is to use the default Kamio et al. method for determining the absolute wavelength scale. If Fe XII λ195.12 has been fit, then the user should set:

```
IDL> fitdata.refwvl=195.119
```

for this line.

For other emission lines, it is necessary to know the wavelength offset between these lines and λ195.12. One option is to take the 'literature' rest wavelengths from Table 2 of Brown et al. (2008, ApJS, 176, 511). For example, the wavelength listed for Fe VIII λ185.21 is 185.213 Å, and that for λ195.12 is 195.119 Å (the same as the value used by Kamio et al.). Therefore, for the Fe VIII fit, the user should just do:

```
IDL> fitdata.refwvl=185.213
```

This assumes that the literature wavelengths – which come from laboratory spectra or previous solar spectra – quoted by Brown et al. have a high accuracy. Warren et al. (2011, ApJ, 727, 58) argue that the EIS spectra themselves can be used to determine rest wavelength offsets. These authors use a quiet Sun offlimb spectrum to measure wavelengths for a number of strong EIS lines, which are shown in their Table 1. For Fe VIII λ185.21 and Fe XII λ195.12 they give wavelengths of 185.2107 and 195.1186 Å, respectively. Placing these on the Kamio et al. wavelength scale (where λ195.12 is at 195.119 Å) implies that the rest wavelength for Fe VIII λ185.21 is 185.2111 Å, which can then be used as the *refwvl* value:

```
IDL> fitdata.refwvl=185.2111
```

A key piece of advice is to make sure that the method by which the velocities are derived is described in detail in the scientific write-up of the work.

5 Technical details

This section gives additional technical details of the software that may be of interest to some users.

5.1 Fitting routines (mpfit)

eis_auto_fit uses the MPFIT fitting routines of Craig Markwardt which are described at:

http://purl.com/net/mpfit

and also in the paper Markwardt (2008, Proc. Astronomical Data Analysis Software and Systems XVIII, Quebec, Canada, ASP Conference Series, Vol. 411, eds. D. Bohlender, P. Dowler & D. Durand, Astronomical Society of the Pacific: San Francisco, p. 251-254).

The specific routine called by *eis_auto_fit* is *mpfitexpr*.

5.2 Specifying the fit function

The MPFIT procedures used for the line fitting require a text string that specifies the fit function. For the present case this function is a linear combination of Gaussians and a straight line for the background. For the case of multi-Gaussian fits the function string is generated by the routine *eis_fit_function*:

```
IDL> expr=eis_fit_function(template)
```

The Gaussian function is *gauss_sg* and the linear function is *line_sg*. These functions are stored in the $SSW/idl/gen/fitting directory. The parameter tying described in Section 2.6 is implemented through the function string.

5.3 The fit template and the mpfit parinfo structure

Constraints on the fit parameters are implemented through the *parinfo* structure that is input to *mpfitexpr*. This structure is generated directly from the *template* structure with:

```
IDL> parinfo=eis_template_parinfo(template)
```

An important feature of *parinfo* that is relevant to fitting the EIS spectral data is the *step* tag. This specifies the step size to be used for each parameter when searching the parameter space for a good solution. By default the MPFIT routines compute the step size automatically based on the size of the initial parameter, however there is a problem for the line centroids. Consider, for example, an emission line at 200 Å. If the initial guess for the centroid is good then the routine should only need to search a region of about ± 0.05 Å in order to find the best centroid value. This is only a range of ±0.025% and, based on the author's experience with fitting, it seems that this can be too small for MPFIT to work properly, leading sometimes to bad fits.

The solution for *eis_auto_fit* is to manually specify the *step* value for *parinfo* for the centroid parameters. Currently a value of 0.005 Å is used. Although less important for the width parameter, a *step* value of 0.005 Å is also used. These values are set with the *eis_template_parinfo* routine.

5.4 Constructing a line profile from the fitdata structure

The *auto_fit* suite of software is set up so that users do not have to directly use the fit parameters stored in the *fitdata* structure, instead the routines *eis_fit_viewer* and *eis_get_fitdata* are available

to enable parameters such as line width and velocity to be extracted in a painless fashion. One circumstance in which a user may want to access the fit parameters directly is when it is required to plot the fitted function, and this serves as an example of how to use the fit parameters stored in *fitdata*.

For a wavelength array stored in the IDL array X, the fit function Y at spatial pixel (i,j) is derived as follows. A single Gaussian function and a linear background are assumed.

```
IDL> aa=fitdata.aa[*,i,j]
IDL> y=aa[0]*exp( - (x-aa[1])^2 / 2. / aa[2]^2 )
IDL> x0=fitdata.x_bg1[i,j]
IDL> x1=fitdata.x_bg2[i,j]
IDL> y0=fitdata.aa[3,i,j]
IDL> y1=fitdata.aa[4,i,j]
IDL> m=(y1-y0)/(x1-x0)
IDL> y = y + m * (x-x0) +y0
```

6 Restrictions

The most common EIS observing sequences that use the narrow slits are:

1. A raster with a single exposure time for each slit position.
2. A sit-and-stare observation with repeated, single exposures at a fixed slit position.

Both of these cases work with *eis_auto_fit*. For the sit-and-stare mode, there is currently a problem whereby *eis_fit_viewer* treats the time dimension as solar-X, but this is simply a display issue and will be fixed in the future.

A less common observing mode is to take multiple exposures at each slit position. For example, the scientist may take a short exposure followed by a long exposure to prevent problems with saturation in strong lines (example EIS studies that use this feature include *nanoflare_sas_v1* and *YKK_EqCHab_01W*). In practical terms, this observation leads to a *windata* structure for which the intensity array has four dimensions (wavelength, solar-X/time, solar-Y, and exposure number). Currently *eis_auto_fit* only fits the arrays belonging to the first exposure. For example, if the exposure sequence is 5s, 10s and 15s, then *eis_auto_fit* only fits the data belonging to the 5s exposure. This problem will be fixed in the future.

An even less common observing mode is when only a single sit-and-stare exposure is taken (for example, the study *ARdiag_sns*). The *windata* intensity structure in this case has three dimensions (wavelength, time, solar-Y), but the time dimension has only one element. Currently the HK spectrum drift correction fails for this case, however, *eis_auto_fit* still fits the data successfully (although without the spectrum drift correction). *eis_fit_viewer* can not be used to view the fits however.

If you find any data-sets that cause *eis_auto_fit* or the associated routines to crash, please contact the author, Peter Young, and give the name of the problem file and the error message.

Case 2: Gaussian fitting for spatially-averaged spectra

Creating spatially-averaged spectra is, at first glance, straightforward: simply average the spectra over the spatial region, and average the errors in quadrature. The situation is complicated for EIS, however, due to the spatial offsets within the EIS instrument which means that a spatial region selected from the Fe XII 195.12 line will not necessarily correspond to the same region, in terms of Y-pixels and even exposure numbers, with another line.

Described below is a set of IDL routines that account for the spatial offsets to create an averaged spectrum that correctly represents the observed spatial structure. The basic steps and corresponding IDL routines are:

1. Create an image for identifying a pixel mask (eis_make_image.pro)
2. Create the pixel mask that defines the spatial region (eis_pixel_mask.pro)
3. Create a 1D spectrum derived from the spatial pixel mask (eis_mask_spectrum.pro)
4. Fit the emission lines (spec_gauss_eis.pro)

We use the following data set as an example:

eis_l0_20061209_113031.fits

which is a 256x256 raster that takes 15 spectral windows with 40s exposures. It is recommended that the user download and calibrate this data with eis_prep:

IDL> eis_prep, l0name, /save, /default, /retain

where l0name is the name of the level-0 FITS file. A level-1 FITS file will be created together with an associated error file.

1 Step 1: make an image

If you already know which wavelength you want to use to identify the spatial structure that you're interested in, do:

IDL> eis_make_image, l1name, 185.21, im185

where l1name is the name of the level-1 FITS file, and 185.21 is the wavelength I'm using for this example. The 2D array im185 contains an image derived from the Fe VIII 185.21 emission line (7 wavelength pixels centered on the 185.21 wavelength have been averaged to generate the image).

Now plot the image using

IDL> plot_image,sigrange(im185)

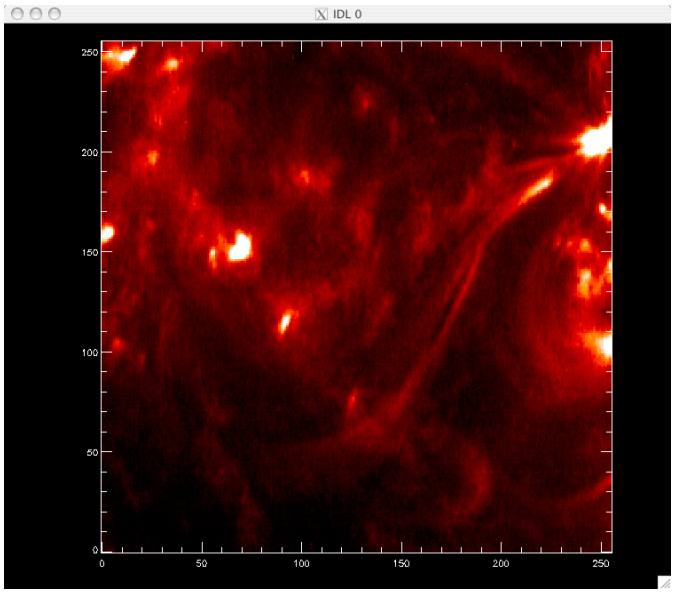

Figure 3. Fe VIII 185.21 image from the 12-Dec-2006 data set.

Since Fe VIII is a cool line (log T = 5.8) then the image consists of a number of fairly small brightenings with a few narrow loop structures.

If you want to browse the data set to choose a different emission line, try using eis_raster_browser:

IDL> eis_raster_browser, l1name

2 Step 2: Create a pixel mask

A pixel mask is an array of same size as the image described earlier but containing only 1's and 0's. Pixels marked with a 1 indicate that they will be used to generate the averaged spectrum.

To create a pixel mask from your image, do

IDL> eis_pixel_mask, im185, mask185, 185.21, 1

where im185 is the image created in the previous section, mask185 will contain the output mask, 185.21 is the wavelength of the line you've specified, and 1 is the size of the slit (should be either 1 for the 1" slit, or 2 for the 2" slit).

Upon calling eis_pixel_mask, you will see the image plotted again (with color table 5) and in the IDL window you will see a menu:

IDL> eis_pixel_mask,im185,mask185,185.21,1

*** CHOOSE A MODE ***
left: polygon mode
middle: painting mode
right: exit

The mouse is used to create a pixel mask by clicking on the image. *You need to use a 3 button mouse for the routine to work!* There are two options for creating the mask:

1. Polygon mode. Click on a number of points in the image and, when complete, the routine will join the dots to create a polygon. The polygon will be filled, and the enclosed pixels will be set to 1 in the pixel mask.
2. Painting mode. By holding down the left mouse button, you can 'paint' over the image, selecting pixels as you go. Selected pixels can be removed by clicking with the middle mouse button.

Generally for small spatial features you will use the Painting mode to accurately select the pixels you need, while Polygon mode is used for choosing large spatial regions.

While clicking on the image, make sure to always check back to the IDL window to see what mode you are currently in. Note that the right mouse button is always used to exit out of a mode.

For our example we are going to choose a small brightening in the image so it is necessary to zoom in on the region of interest:

```
IDL> eis_pixel_mask,im185,mask185,185.21,1,xrange=[50,100],yrange=[120,170], col=255
```

which displays X-pixel range 50 to 100 and Y-pixel range 120 to 170. Note that col=255 is used to set the color of the over-plotted pixels to white.

Use the Painting mode to select the pixels. In Figure 4 the selected pixels are displayed in white.

If you want to reset the mask, then simply exit out of the routine and do:

```
IDL> mask185=0
```

Section 5 of this document explains how the Polygon option in *eis_pixel_mask* works.

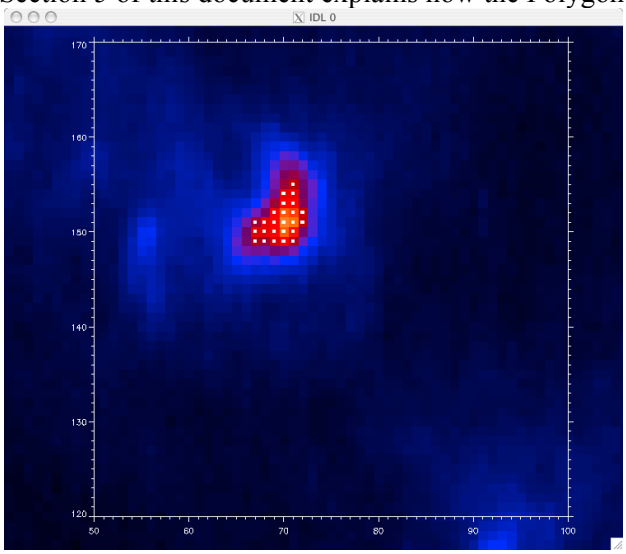

Figure 4. Fe VIII image showing the selected pixels (small white squares).

2.1 Viewing the pixel map in different wavelengths

It is useful to check how the pixel mask looks for other wavelengths. To create the mask for, e.g., Si VII 275.35, do

IDL> mask275=eis_adjust_mask(mask185,275.35,time=obs_time)

eis_adjust_mask is an important routine as it takes into account the various spatial offsets to move the selected pixels to spatially match the original wavelength.

The observation time needs to be input as the X-offset between the two detectors changed on 24-Aug-2008. No other time effects are currently known.

To give an indication of the shifts involved in the present case, do:

IDL> plot_image,mask185.image+mask275.image,xra=[50,100],yra=[120,170]

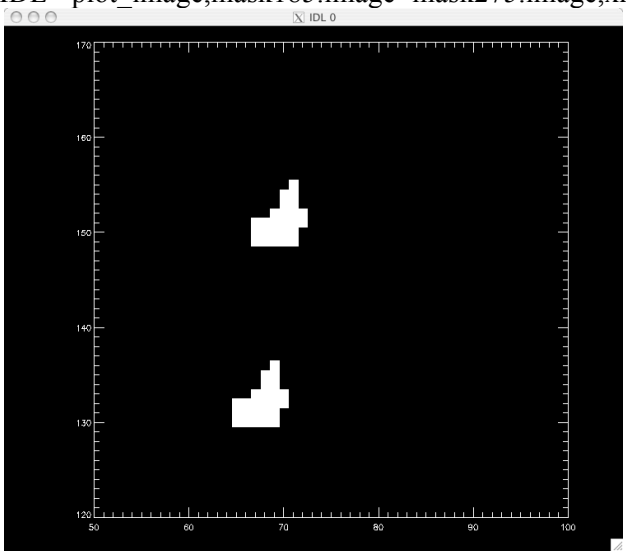

Figure 5. Pixel masks for Fe VIII 185 (upper) and Si VII 275 (lower).

The upper set of pixels are those from the original Fe VIII 185.21 mask, while the lower set of pixels are those from the Si VII 275.35 mask. The latter is seen to be shifted in both X and Y.

Now since Si VII and Fe VIII are formed at the same temperature, then we are able to check how accurately the derived Si VII mask sits on the brightening seen in the Si VII image (remember that mask275 has been derived from the mask chosen in the Fe VIII image).

We first create a 275 image:

IDL> eis_make_image, l1name, 275.35, im275

and then do:

IDL> eis_pixel_mask,im275,mask275,275.35,1,col=255,xrange=[50,100],yrange=[120,170]

which gives the image shown in Figure 6. The match is fairly good although seems to be off by about 1 pixel in the Y-direction. Since the shift of the pixel mask is done in integer pixel

numbers, then this is within the uncertainty of the method taking into account also the uncertainty in the measured EIS spatial offsets.

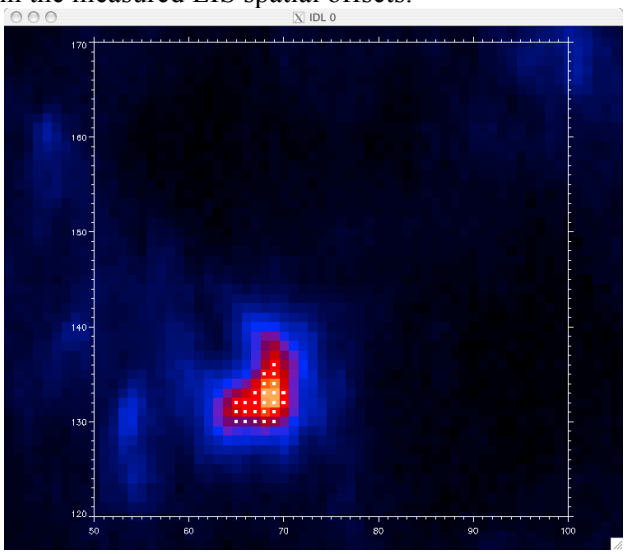

Figure 6. Si VII 275.35 image with pixel mask overplotted.

3 Deriving the averaged spectrum

Once you are happy with your chosen pixel mask, an EIS spectrum averaged over the spatial region can be derived by doing:

```
IDL> eis_mask_spectrum, l1name, mask185, swspec=swspec, lwspec=lwspec
```

swspec and lwspec are both structures with the following tags:

```
IDL> help,swspec,/str
** Structure <1539eaa4>, 5 tags, length=36868, data length=36866, refs=1:
   WVL             DOUBLE    Array[2048]
   INT             FLOAT     Array[2048]
   ERR             FLOAT     Array[2048]
   QUAL            INT       Array[2048]
   QUAL_MAX        INT           25
```

You will see that the spectrum is defined for the full size of the CCD (2048 pixels). You can view it by doing:

```
IDL> plot,swspec.wvl,swspec.int,psym=10,/xsty
```

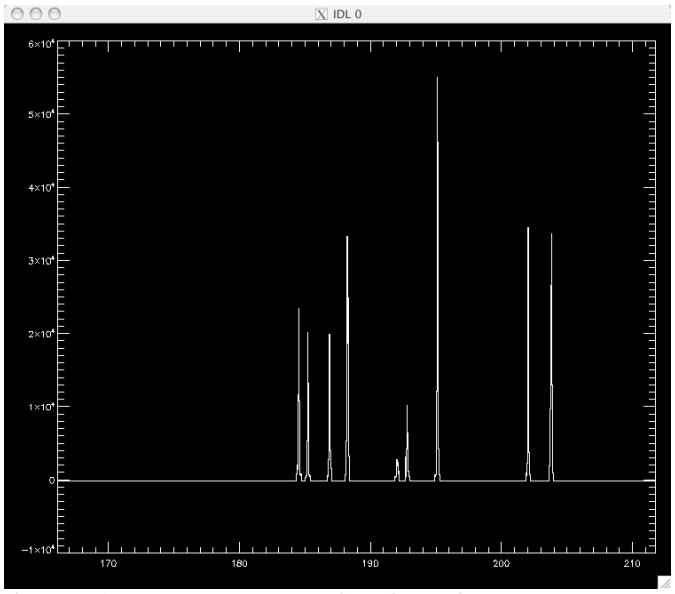

Figure 7. Averaged spectrum derived from pixel mask.

You'll see only a few emission lines that correspond to the wavelength windows of the EIS raster. All other wavelengths are set to an intensity value of -100.

Note that the number of pixels averaged over to generate the spectrum is stored in the tag QUAL_MAX. The significance of the QUAL tag is discussed in the separate tutorial on SPEC_GAUSS_WIDGET, the Gaussian fitting routine.

3.1 Tilt and orbit corrections

The slit tilt and spectrum drift will mean that averaging spectra over a spatial area will lead to emission lines being artificially broadened due to the varying positions of the line centroids on the detector. For small spatial areas, say 10x10 pixels, this will be a small effect but for large areas the broadening may be significant. This is particularly so for the 2" slit which has a larger tilt than the 1" slit.

To prevent this broadening from occurring, the keyword /shift should be set in conjunction with the /refill keyword:

IDL> eis_mask_spectrum,l1name,mask185,/shift,/refill,swspec=swspec,lwspec=lwspec

The /shift keyword forces the individual windata structures for each wavelength to be processed with the routine *eis_shift_spec*. This routine interpolates the individual pixel spectra onto the same wavelength scale by making use of the *wave_corr* tag present within the windata structures (see Section 1.6 for more details).

Missing pixels are a nuisance for *eis_shift_spec* as they essentially lead to a doubling of the number of missing pixels in the arrays (interpolation between a good pixel and a missing pixel leads to both being set to missing). For this reason it is recommended that the /refill option always be used when /shift is set, as it greatly reduces the number of missing pixels.

4 Step 4: Fitting Gaussians

Gaussians can be fit to the output spectrum using the routine SPEC_GAUSS_EIS, which is called as:

IDL> spec_gauss_eis, swspec

it is actually a wrapper to a general purpose Gaussian fitting routine called SPEC_GAUSS_WIDGET present in the /gen branch of Solarsoft. A tutorial explaining how to use 'spec_gauss_widget' is available in Solarsoft at:

$SSW/gen/idl/fitting/spec_gauss_widget_tutorial.pdf

The tutorial is also available through the EIS wiki.

5 The Polygon option in eis_pixel_mask

Start up eis_pixel_mask on the whole Fe VIII 185 image:

IDL> eis_pixel_mask, im185, mask185, 185.21, 1, col=255

and click somewhere on the image with the left mouse button – this puts the routine into Polygon mode.

I will choose an approximately rectangular region in the bottom left of the image by clicking three times with the left mouse button. You'll see that after the first button press, subsequent ones will show a line joining the newly added point to the previous. Figure 8 shows the result after 3 button presses.

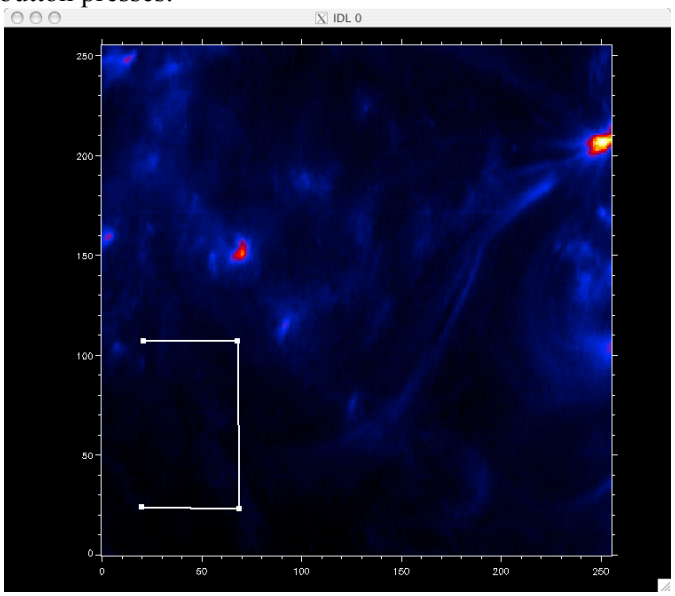

Figure 8. An image from during the process to select a region with the Polygon option in eis_pixel_mask.

To 'close' the rectangle, simply click on the RIGHT mouse button, which exits the Polygon mode and joins up the last two dots. Figure 9 shows the result where the selected pixels form a rectangular block in the image.

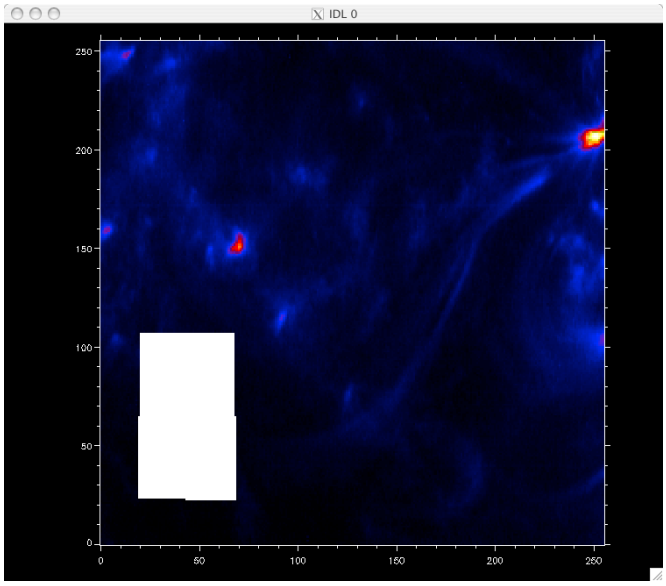

Figure 9. The selected spatial region after using the Polygon option in eis_pixel_mask is shown.

To fine-tune the selected block of pixels, the Painting mode can be used.

6 Special options

6.1 The /fix keyword

A common feature of recent EIS data are occasional exposures that are anomalously bright. These are not flagged by *eis_prep* as bad data, and so they can mess up the output of *eis_mask_spectrum*. Generally these exposures can be treated by giving the keyword /fix. This calls out to the routine *eis_fix_windata.pro* to set the exposures to missing. See the header for *eis_fix_windata* for more details on this procedure.

6.2 The /sum keyword

The default behavior of eis_mask_spectrum is to average the spectra over the spatial pixels identified in the pixel mask. By setting the /sum keyword, the spectra are instead summed. An example of using this option is from Young et al. (2013, ApJ, 766, 127) who considered that a flare kernel was not resolved by the instrument and so the observed spatial extent of the kernel was due to the point spread function of the telescope. In this case the intensity should be summed over the kernel's extent, rather than averaged.

6.3 The subtract_l1name= keyword

Young et al. (2013, ApJ, 766, 127) presented observations of a flare kernel. They created a background-subtracted spectrum of the flare kernel using eis_mask_spectrum. The background subtraction was performed by taking the same spatial location from the previous raster. This was done with the keyword subtract_l1name:

IDL> eis_mask_spectrum,file,mask,swspec=swspec,lwspec=lwspec,subtract_l1name=prev_file

prev_file must be the name of a level-1 file.

The background subtraction is done on a pixel-by-pixel basis, and then the pixel mask is applied to yield the averaged spectrum. This method was needed because the background plasma was resolved by EIS, whereas the flare kernel was not.

7 Restrictions

The routine *eis_mask_spectrum* is only intended to work for spatial rasters. Potentially it could work on sit-and-stare data, but this is not tested (!).

Some spatial rasters have multiple exposures at each slit position and these can be processed by *eis_mask_spectrum*. Note that the routine checks the exposure times of each exposure in order to implement the following behavior: exposures with exposure times within 10% of the maximum exposure time, will be included in the spectrum sum, otherwise they will be ignored. For example, if the exposure times are 5s and 60s then the 5s exposure will be ignored and only the 60s exposure will be used. If the exposure times are both 60s, then the two exposures will be summed. The reason for this behavior is that a short exposure will have significantly worse signal-to-noise than a long exposure, and so combining it with the long exposure will degrade the quality of the long exposure spectrum.

In some cases the user may want to only use the short exposure, for example if the long exposure is badly saturated. The keyword iexp= can be used to select a particular exposure. (First run eis_mask_spectrum without the iexp keyword, and you will see a list of the different exposure times and indices from which you can select.)

Appendix

1 eis_getwindata operations

A separate Software Note (No. 21, "The windata routines for EIS data analysis") now describes the various routines that operate on the windata structure. Please refer to this.

2 Document update history

Version 2.6: modified Appendix 1 to reference the new Software Note No. 21.

Version 2.5: added Section 6 (eis_mask_spectrum); changed the xsc and ysc keywords for eis_pixel_mask to xrange and yrange; added text about iexp= to Section 7.

Version 2.4: added Section 5.4 and expanded Sections 1.2 and 2.5.

Version 2.3: added Section 1.5 – common options for using eis_auto_fit.

Version 2.2: modified Section 3.1 as the calling sequence for eis_mask_spectrum has changed (now use /shift instead of offset=). Added sections on restrictions for using the routines (both Case 1 and Case 2). Added section on *eis_shift_spec*.

Version 2.1: added section about eis_combine_sitstare_windata (Appendix 1.4) and corrected an error in the eis_trim_windata section (Appendix 1.3). Added new sections to Case 1, Section 5 (Technical Details). Also performed some minor corrections in various parts of the text.

Version 2: since the release of the new Kamio et al. method for dealing with the spectrum drift and the subsequent storing of the spectrum drift in the windata.wave_corr tag, there is no need to call eis_wave_corr and input the offset= keyword for each routine. Significant changes to the text have been made to Section 1 to reflect this. Section 4 on absolute velocities has been added and Sect. 2.3 has been modified. Document has now become EIS Software Note #16.